Exhibit A

Universal Description, Discovery and Integration

UDDI Technical White Paper

September 6, 2000

Abstract

Universal Description, Discovery and Integration (UDDI) is a specification for distributed Web-based information registries of Web services. UDDI is also a publicly accessible set of implementations of the specification that allow businesses to register information about the Web services they offer so that other businesses can find them.

Web services are the next step in the evolution of the World Wide Web (WWW) and allow programmable elements to be placed on Web sites where others can access distributed behaviors. UDDI registries are used to promote and discover these distributed Web services. This paper describes the capabilities that these registries add to the World Wide Web.

The intended audience is the anyone looking for a conceptual overview of UDDI for the purpose of understanding what it is, who uses it, and how a distributed registry makes it possible for your programs to discover and interact with Web services that other companies expose on the Web

Introduction

Overview

The Universal Description, Discovery and Integration (UDDI) specifications define a way to publish and discover information about Web services. The term "Web service" describes specific business functionality exposed by a company, usually through an Internet connection, for the purpose of providing a way for another company or software program to use the service.

Web services are becoming the programmatic backbone for electronic commerce. For example, one company calls another's service to send a purchase order directly via an Internet connection. Another example is a service that calculates the cost of shipping a package of a certain size or weight, so many miles via a specific carrier.

At first glance, it would seem simple to manage the process of Web service *discovery*. After all, if a known business partner has a known electronic commerce gateway, what's left to discover? The tacit assumption, however, is that all of the information is already known. When you want to find out which business partners have which services, the ability to discover the answers can quickly become difficult. One option is to call each partner on the phone, and then try to find the right person to talk with. For a business that is exposing Web services, having to staff enough highly technical people to satisfy random discovery demand is difficult to justify.

Another way to solve this problem is through an approach that uses a Web services description file on each company's Web site. After all, Web crawlers work by accessing a registered URL and are able to discover and index text found on nests of Web pages. The "robots.txt" approach, however, is dependent on the ability for a crawler to locate each Web site and the location of the service description file on that Web site. This distributed approach is potentially scalable but lacks a mechanism to insure consistency in service description formats and for the easy tracking of changes as they occur.

UDDI takes an approach that relies upon a distributed registry of businesses and their service descriptions implemented in a common XML format.

UDDI business registrations and the UDDI business registry

The core component of the UDDI project is the UDDI business registration, an XML file used to describe a business entity and its Web services. Conceptually, the information provided in a UDDI business registration consists of three components: "white pages" including address, contact, and known identifiers; "yellow pages" including industrial categorizations based on standard taxonomies; and "green pages", the technical information about services that are exposed by the business. Green pages include references to specifications for Web services, as well as support for pointers to various file and URL based discovery mechanisms if required.

Using UDDI

UDDI includes the shared operation of a business registry on the Web. For the most part, programs and programmers use the UDDI Business Registry to locate information about services and, in the case of programmers, to prepare systems that are compatible with advertised Web services or to describe their own Web services for others to call. The UDDI Business Registry can be used at a business level to check whether a given partner has particular Web service interfaces, to find companies in a given industry with a given type of service, and to locate information about how a partner or intended partner has exposed a Web service in order to learn the technical details required to interact with that service.

After reading this paper, the reader will have a clearer understanding of the capabilities defined in the UDDI specifications and have a clearer understanding of the role of Web service registries that implement these specifications.

Background

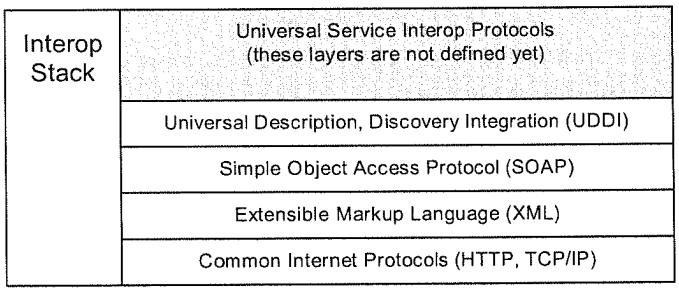
The number of ways that companies are using the World Wide Web varies considerably. Many companies are starting to define ways to allow their internal applications to interact with the business systems at other companies using the emerging Web infrastructure. Left alone, each company invents a unique approach based on the experiences of designers, available technologies, and project budgets. The proliferation of integration approaches and unique solutions have spawned an entire sub-industry focused on bridging incompatible service layers within and across company boundaries.

Recent work within the W3C starts to raise hopes that Extensible Markup Language (XML) will play a role in simplifying the exchange of business data between companies. Further, collaboration between computer industry giants and small companies alike have outlined a framework called SOAP that allows one program to invoke service interfaces across the Internet, without the need to share a common programming language or distributed object infrastructure. All of this is good news for companies feeling the cost pressures associated with electronic commerce because the foundations for common interoperability standards are being laid. Because of these foundation technologies and emerging standards, some of the intractable problems of the past are becoming easier to approach.

From XML and SOAP, one can observe that the integration and interoperability problem has been simplified in layers. XML provides a cross-platform approach to data encoding and formatting. SOAP, which is built on XML, defines a simple way to package information for exchange across system boundaries. SOAP bindings for HTTP are built on this packaging protocol and define a way to make remote procedure calls between systems in a manner that is independent of the programming language or operating system choices made by individual companies. Prior approaches involved complex distributed object standards or technology bridging software. Neither of these approaches has proven to be cost effective in the long run. Using XML and SOAP, this cross-language, cross-platform approach simplifies the problem of making systems at two companies compatible with each other.

Even when one considers XML and SOAP, though, there are still vast gaps through which any two companies can fall in implementing a communications infrastructure. As any industry pundit will tell you: "What is required is a full end-to-end solution, based on standards that are universally supported on every computing platform." Clearly, there is more work to do to achieve this goal. The UDDI specifications borrow the lesson learned from XML and SOAP to define a next-layer-up that lets two companies share a way to query each other's capabilities and to describe their own capabilities.

The following diagram depicts this layered view:



UDDI is a "next layer" in an emerging *stack* enabling rich Web services. UDDI uses standards-based technologies such as TCP/IP, HTTP, XML and SOAP to create a uniform service description format and service discovery protocol.

Figure 1

UDDI – the technical discovery layer

The Universal Description, Discovery and Integration (UDDI) specification describes a conceptual cloud of Web services and a programmatic interface that define a simple framework for describing any kind of Web service. The specification consists of several related documents and an XML schema that defines a SOAP-based programming protocol for registering and discovering Web services. These specifications were defined over a series of months by technicians and managers from several leading companies. Together, these companies have undertaken the task of building the first implementation

of the UDDI services and running these services as a publicly accessible, multi-site partnership that shares all registered information.

The following diagram shows the relationship between the specifications, the XML schema and the UDDI business registry cloud that provides "register once, published everywhere" access to information about Web services.

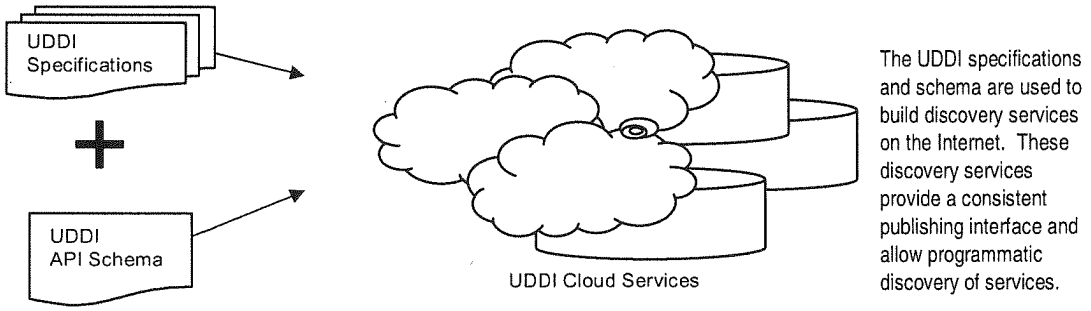


Figure 2

Using the UDDI discovery services, businesses individually register information about the Web services that they expose for use by other businesses. This information can be added to the UDDI business registry either via a Web site or by using tools that make use of the programmatic service interfaces described in the UDDI Programmer's API Specification. The UDDI business registry is a logically centralized, physically distributed service with multiple root nodes that replicate data with each other on a regular basis. Once a business registers with a single instance of the business registry service, the data is automatically shared with other UDDI root nodes and becomes freely available to anyone who needs to discover what Web services are exposed by a given business.

Next steps

As the layers in figure 1 show, it is important to note that UDDI does not form a full-featured discovery service. UDDI services are targeted at enabling technical discovery of services. With the facilities defined by UDDI, a program or programmer can locate information about services exposed by a partner, can find whether a partner has a service that is compatible with in-house technologies, and can follow links to the specifications for a Web service so that an integration layer can be constructed that will be compatible with a partners service. Businesses can also locate potential partners through UDDI directly, or more likely, from online marketplaces and search engines that use UDDI as a data source for their own value-added services. Technical compatibility can be discovered so that software companies can use the UDDI registries on the Web to automatically configure certain technical connections as software is installed or accounts are configured.

Business discovery and UDDI

UDDI is designed to complement existing online marketplaces and search engines by providing them with standardized formats for programmatic business and service discovery. The ability to locate parties that can provide a specific product or service at a given price or within a specific geographic boundary in a given timeframe is not directly covered by the UDDI specifications. These kinds of advanced discovery features require further collaboration and design work between buyer and sellers. Instead, UDDI forms the basis for defining these services in a higher layer.

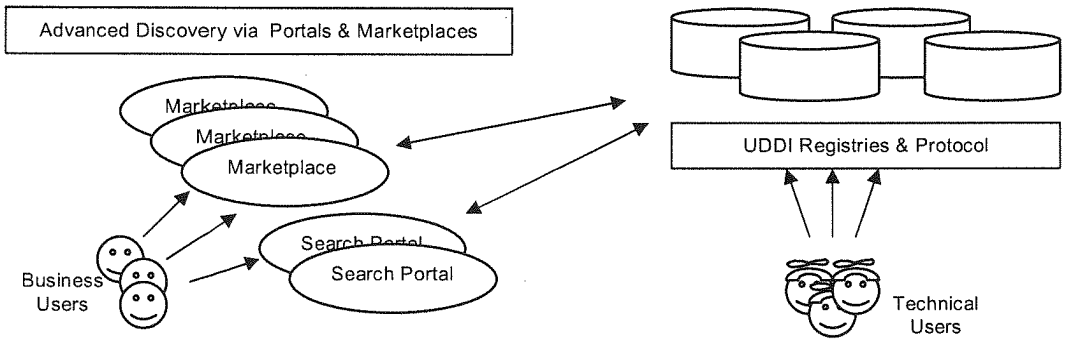


Figure 3

In Figure 3 we can see the relationship between the technical discovery layers defined by UDDI and the role of aggregation and specialized search capabilities that address business level searches. Currently, marketplaces and search portals fill this need, and can be integrated or populated using information published in the UDDI distributed registries.

Future work

The teams working on the UDDI are planning on extending the functionality beyond what is in the Open Draft specification to address more than just technical discovery. Future features will address the ability to locate products and services, define Web service implementation conventions and provide the ability to manage hierarchical business organizations, communities and trade groups. The driving goal is to provide a public specification for Web service interoperability, whether the focus is marketplace-to-marketplace or business-to-business.

The remainder of this paper is a technical overview of the various features of the UDDI discovery service and specifications.

Technical overview

The Universal Description, Discovery and Integration (UDDI) specifications consist of an XML schema for SOAP messages, and a description of the UDDI API specification. Together, these form a base information model and interaction framework that provides the ability to publish information about a broad array of Web services.

Four information types

The core information model used by the UDDI registries is defined in an XML schema. XML was chosen because it offers a platform-neutral view of data and allows hierarchical relationships to be described in a natural way. The emerging XML schema standard was chosen because of its support for rich data types as well as its ability to easily describe and validate information based on information models represented in schemas.

The UDDI XML schema defines four core types of information that provide the kinds of information that a technical person would need to know in order to use a partners Web services. These are: business information; service information, binding information; and information about specifications for services.

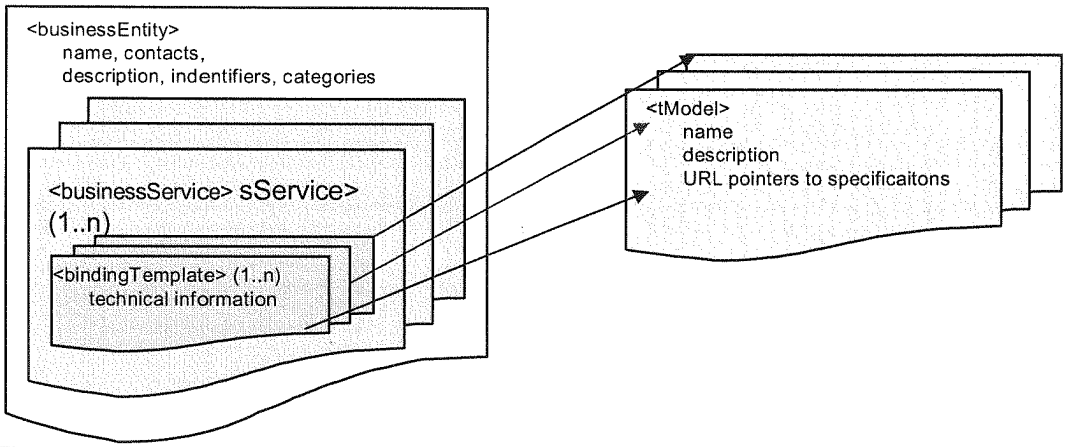


Figure 4

The information hierarchy and the key XML element names that are used to describe and discover information about Web services are shown in figure 4[1].

Business information: the businessEntity element

Many partners will need to be able to locate information about your services and will have as starting information a small set of facts about your business. Technical staff, programmers or application programs themselves will know either your business name or perhaps your business name and some key identifiers[2], as well as optional categorization and contact information. The core XML elements for supporting publishing and discovering information about a business – the UDDI Business Registration -- are contained in a structure named "businessEntity"[3]. This structure serves as the top-level information manager for all of the information about a particular set of information related to a business unit[4].

The overall businessEntity information includes support for "yellow pages" taxonomies so that searches can be performed to locate businesses who service a particular industry or product category, or who are located within a specific geographic region.

Service information: the businessService and bindingTemplate elements

Technical and business descriptions of Web services – the "green pages" data -- live within sub-structures of the businessEntity information. Two structures are defined: businessService and bindingTemplate. The businessService structure is a descriptive container that is used to group a series of related Web services related to either a business process or category of services. Examples of business processes that would include related Web service information include purchasing services, shipping services, and other high-level business processes.

These businessService information sets can each be further categorized – allowing Web service descriptions to be segmented along combinations of industry, product and service or geographic category boundaries.

Within each businessService live one or more technical Web service descriptions. These contain the information that is relevant for application programs that need to connect to and then communicate with a remote Web service. This information includes the address to make contact with a Web service, as

---

[1] See appendix A for a more complete view of the UDDI information model.

[2] Business identifiers can include D&B numbers, tax numbers, or other information types via which partners will be able to identify a business uniquely.

[3] See the UDDI XML schema –http://www.uddi.org/schema/uddi_1.xsd

[4] Complex business unit information should be registered in separate businessEntity records.

well as support for option information that can be used to describe both hosted services[5] and services that require additional values to be discovered prior to invoking a service[6]. Additional features are defined that allow for complex routing options such as load balancing[7].

Specification pointers and technical fingerprints

The information required to actually invoke a service is described in the information element named bindingTemplate. This was described in the previous section. However, it is not always enough to simply know where to contact a particular Web service. For instance, if I know that a business partner has a Web service that lets me send them a purchase order, knowing the URL for that service is not very useful unless I know a great deal about what format the purchase order should be sent in, what protocols are appropriate, what security required, and what form of a response will result after sending the purchase order. Integrating all parts of two systems that interact via Web services can become quite complex.

As a program or programmer interested in specific Web services, information about compatibility with a given specification is required to make sure that the right Web service is invoked for a particular need. For this reason, each bindingTemplate element contains a special element that is a list of references to information about specifications. Used as an opaque[8] set of identifiers, these references form a technical fingerprint that can be used to recognize a Web service that implements a particular behavior or programming interface.

In our purchase order example, the Web service that accepts the purchase order exhibits a set of well-defined behaviors if the proper document format is sent to the proper address in the right way. A UDDI registration for this service would consist of an entry for the business partner, a logical service entry that describes the purchasing service, and a bindingTemplate entry that describes the purchase order service by listing its URL and a reference to a tModel.

These references are actually the keys that can be used to access information about a specification. Called "tModels", this information is *metadata* about a specification, including its name, publishing organization, and URL pointers[9] to the actual specifications themselves. In our example, the tModel reference found in the bindingTemplate is a pointer to information about the specifics of this purchase order Web service. The reference itself is a pledge by the company that exposes the Web service that they have implemented a service that is compatible with the tModel that is referenced. In this way, many companies can provide Web services that are compatible with the same specifications.

The programmer's API

The UDDI specifications include definitions for Web service interfaces that allow programmatic access to the UDDI registry information. The full definition of the programmer's API is found in the Programmer's API Specification document. The API capabilities are briefly discussed below.

The API is divided into two logical parts. These are the Inquiry API and the Publishers' API. The Inquiry API is further divisible into two parts – one part used for constructing programs that let you search and browse information found in a UDDI registry, and another part that is useful in the event that Web service invocations experience failures. Programmers can use the Publishers API to create rich

---

[5] Other companies provision hosted services, typically on a fee base. Marketplaces are good examples of hosted services.

[6] Software packages are a good example of this kind of requirement. Individual installations of a given package may have specific values that must be accommodated prior to connecting with a service.

[7] The hostingRedirector feature allows both hosting and other redirection capabilities to be deployed. See the appendix on redirection in the UDDI programmers API specification for more details.

[8] The term *opaque* in this context alludes to the fact that simply knowing a key value for a particular specification is equivalent to knowing a service is compatible with that specification.

[9] **Note**: tModels do not actually contain the actual specifications. UDDI defines a framework for taking advantage of URLs and web servers, so that individual organizations can maintain centralized specifications. Individual implementations can then be located based on whether or not they contain references to specific specification keys.

interfaces for tools that interact directly with a UDDI registry, letting a technical person manage the information published about either a businessEntity or a tModel structure.

Built on SOAP

The Simple Object Access Protocol (SOAP) is a W3C draft note describing a way to use XML and HTTP to create an information delivery and remote procedure mechanisms. Several companies, including IBM, Microsoft, DevelopMentor and Userland Software, submitted this draft note to the W3C for the purpose of, among other things, standardizing RPC (simple messaging) conventions on the World Wide Web. In its current state, the draft note describes a specification that is useful for describing a Web service. The companies that collaborated on UDDI decided to base the UDDI APIs on this SOAP specification. The specifics of how SOAP and XML are used by UDDI registry *Operators* are defined in the appendices in the API specification itself.

All of the API calls defined by the UDDI Programmer's API Specification behave synchronously – and all of the distributed UDDI registry *Operator Sites* support all of the calls described in the Programmer's API Specification.

The Inquiry API

The Inquiry API consists of two types of calls that let a program quickly locate candidate businesses, Web services and specifications, and then drill into specifics based on overview information provided in initial calls. The APIs named *find_xx* provide the caller with a broad overview of registration data based on a variety of search criteria. Alternately, if the actual keys of specific data are known ahead of time, up to date copies of a particular structure (e.g. businessEntity, businessService, bindingTemplate, tModel) can be retrieved in full via a direct call. These direct calls are called the *get_xx* APIs.

The UDDI invocation model

Each individual advertised Web service is modeled in a bindingTemplate structure. Invocation of a Web service is typically performed based on cached bindingTemplate data. With this in mind, the general scenario for using UDDI becomes clear when you consider the preparation required to write a program that uses a specific Web service. The following recipe outlines these steps

1. The programmer, chartered to write a program that uses a remote Web service, uses the UDDI business registry (either via a Web interface or other tool that uses the Inquiry API) to locate the businessEntity information registered by or for the appropriate business partner that is advertising the Web service.
2. The programmer either drills down for more detail about a businessService or requests a full businessEntity structure. Since businessEntity structures contain all information about advertised Web services, the programmer selects a particular bindingTemplate[10] and saves this away for later use.
3. The programmer prepares the program based on the knowledge of the specifications for the Web service. This information may be obtained by using the tModel key information contained in the bindingTemplate for a service.
4. At runtime, the program invokes the Web service as planned using the cached bindingTemplate information (as appropriate).

In the general case, assuming the remote Web service and the calling program each accurately implement the required interface conventions (as defined in the specification referenced in the tModel information), the calls to the remote service will function successfully. The special case of failures and recovery is outlined next.

---

[10] Using the find_xx inquiry API, a UDDI compatible browser can display more or less detail as someone searches through information. Once the appropriate information is located, the get_xx call returns full information about one of the four key UDDI XML structures.

Recovery after remote Web service call failure

One of the key benefits of maintaining information about Web services in a distributed UDDI Registry is the "self service" capability provided to technical personnel. The recipe in the previous section outlined the tasks that the programmer is able to accomplish using the information found in the UDDI registry. This is all fine and well, but additional benefits are possible. These benefits of using a distributed UDDI registry with information hosted at an *Operator Site* are manifested in disaster recovery scenarios.

Web services businesses using Web services to do commerce with their partners need to be able to detect and manage communication problems or other failures. A key concern is the inability to predict, detect, or recover from failures within the systems of the remote partner. Even simple situations such as temporary outages caused by nightly maintenance or back-ups can make the decision to migrate to Web services difficult.

On the other hand, if you are the company that makes direct Web service connections possible, disaster recovery and the ability to migrate all of your business partners to a back-up system are prime concerns.

UDDI starts to address these "quality of service" issues by defining a calling convention that involves using cached bindingTemplate information, and when failures occur, refreshing the cached information with current information from a UDDI Web registry. The recipe for this convention goes like this:

1. Prepare program for Web service, caching the required bindingTemplate data for use at run-time.
2. When calling the remote Web service, use the cached bindingTemplate data that was obtained from a UDDI Web registry.
3. If the call fails, use the bindingKey value and the get_bindingTemplate API call to get a fresh copy of a bindingTemplate for this unique Web service.
4. Compare the new information with the old – if it is different, retry the failed call. If the retry succeeds, replace the cached data with the new data.

Behind the scenes, when a business needs to redirect traffic to a new location or backup system, they only need to activate the backup system and then change the published location information for the effected bindingTemplates. This approach is called *retry on failure* – and is more efficient than getting a fresh copy of bindingTemplate data prior to each call.

The Publication API

The Publication API consists of four *save_xx* functions and four delete_xx functions, one each for the four key UDDI data structures (businessEntity, businessService, bindingTemplate, tModel). Once authorized, an individual party can register any number of businessEntity or tModel information sets, and can alter information previously published. The API design model is simple – changes to specific related information can be made and new information be saved using save. Complete structure deletion is accommodated by the delete calls. See the Programmer's API Specification for more information on this topic.

Security: Identity and authorization

The key operating principal for the UDDI Publishers' API is to only allow authorized individuals to publish or change information within the UDDI business registry. Each of the individual implementations of the distributed UDDI business registry maintains a unique list of authorized parties and tracks which businessEntity or tModel data was created by a particular individual. Changes and deletions are only allowed if a change request (via API call) is made by the same individual who created the effected information.

Each instance of a UDDI business registry, called an *Operator Site,* is allowed to define its own end user authentication mechanism[11], but all of the contracted UDDI *Operator Sites* are required to meet certain minimum criteria that provide similar security protections.

Other information

For more details on the UDDI schema or the UDDI Programmer's API Specification, consult the documents that are available on the uddi.org Web site.

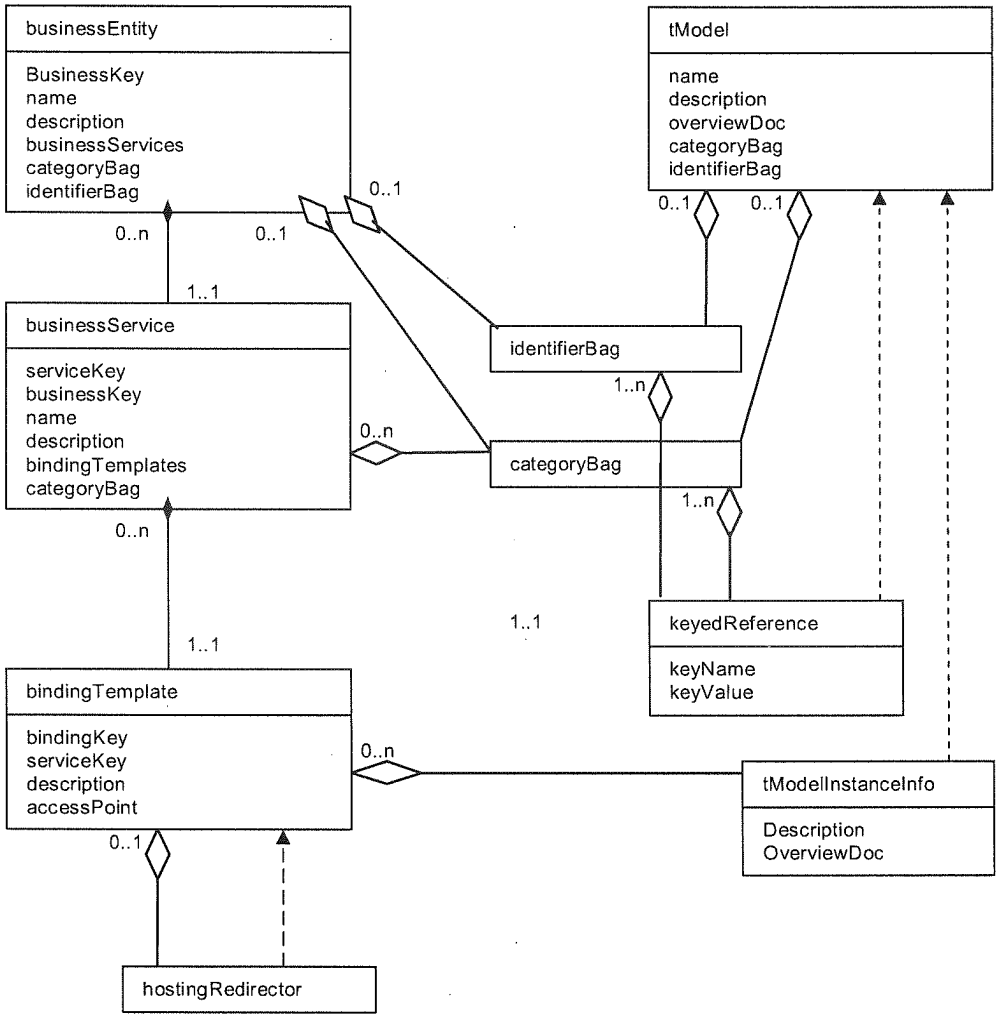
These specifications are published as a set of linked HTML documents, with specific tModels defined for related sub service offerings. The individual tModel references are actually overview information with shared links to overview, API call, and appendix information within the overall specification.

Printable versions of the full specification in Microsoft Word and PDF format will also be available for download.

[11] Private implementations that are built using the UDDI specifications cannot be forced to implement any specific conventions or requirements. For this reason, be sure to consult with the UDDI implementer if you have questions about security or information access control policies.

Appendix A: UDDI information model

The diagram below shows the relationships between the different core information elements that make up the UDDI information model.



Resources

This section contains the locations of various specifications, document references and useful information where you can learn more about this subject.

- W3C XML and related recommendations: http://www.w3.org/TR
- SOAP 1.1 W3C note: http://www.w3.org/TR/#Notes
- UDDI Web site: http://www.uddi.org
    - UDDI Programmer's API Specification: http://www.uddi.org/pubs/UDDI_Programmers_API_Specification.pdf
    - UDDI Data Structure Reference: http://www.uddi.org/pubs/UDDI_XML_Structure_Reference.pdf
    - UDDI Revision 1.0 schema: http://www.uddi.org/schema/uddi_1.xsd